# A Software Reuse in Small Scale Industry: A Survey

Tincy Rani, Sushil Garg
*CES Dept, RIMT-IET, Mandi Gobindgarh, HOD in CSE Dept, RIMT-IET, Mandi Gobindgarh.*

## Abstract

*Software Engineering is the very integrated part of the industry from mid nineties. Small scale enterprises are a very important in the gears of the world economy. Software industry becomes significant economical activity from the last few decades. According to Fayad et al. [8] 99.2% of software development companies are small (fewer than 150 employees).They develop significant products, for the construction of which the firms need efficient software engineering practices that are suitable for their particular size and type of business. Effectively gathering user requirements is a critical first step of any project and perhaps one of the most challenging project management skills. To avoid cost overruns, dissatisfied users or even project cancellation, it is vitally important to build the project on well-formed, testable, and verifiable user requirements. After project requirement and implementation by firm implementation of software is comes into act which in turn complete the project in major sense. Usually companies tends to reuse their assets, resources and software policies to cut-off their expenses and to increase utilization of available resources.*

## 1. Introduction

Software reuse is the process of creating software systems from existing software rather than building software systems from scratch. This simple yet powerful vision was introduced in 1968. Software products are expensive. Therefore, software project managers are always worried about the high cost of software development. A possible way to reduce development cost is to reuse parts from previously developed software. In addition to reduced development cost and time, Reuse also lead to higher quality of the developed products since the reusable components are ensured to have high quality. A reuse approach that is of late gaining prominence is component based development. Component-based software development is different from the traditional software development in that software is developed by assembling software from off-the-shelf components. Reuse is an umbrella concept, encompassing a variety of approaches and situations [Morisio02]. The reusable components or assets can take several forms: subroutines in library, free-standing COTS (Commercial-Off-The-Shelf) or OSS (Open Source Software) components, modules in a domain-specific framework (e.g. Smalltalk MVC classes), or entire software architectures and their components forming a product line or a product family.

Morisio02 et al. define reusability as a combination of two characteristics:

1. Usefulness, which is the extent to which an asset is often needed.
2. Usability, which is the extent to which an asset is packaged for reuse.

Frakes et al [3] define software reuse as “The use of existing software knowledge or artifacts to build new software artifacts”, a definition that includes reuse of software knowledge. Morisio’s [2] definition is closer to what is meant by “software reuse” in our research; i.e. reuse of building blocks knowledge, or patterns may happen without reuse of building blocks and is captured in domain engineering. Developing for reuse has its price, which is the reason for analyzing the success of reuse programs to improve the chances of succeeding.

## 2. Basic issues in any reuse program

The following are some of the basic issues that must be clearly understood for starting any reuse program.

### 2.1. Component creation

For component creation, the reusable components have to be first identified. Selection of the right kind of components having potential for reuse is important. Domain analysis as a promising technique which can be used to create reusable components.

### 2.2. Component indexing and storing

Indexing requires classification of the reusable components so that they can be easily searched when we look for a component for reuse. The components need to be stored in a relational database management system (RDBMS) or an Object-Oriented Database System (ODBMS) for efficient access when the number of components becomes large.

### 2.3. Component searching

The programmers need to search for right components matching their requirements in database of components. To be able to decide whether they can reuse the component. To facilitate understanding, the components should be well documented and should do something simple.

### 2.4. Component adaptation

Often, the components may need adaptation before they can be reused, since a selected component may not exactly fit the problem at hand. However, tinkering with the code is also not a satisfactory solution because this is very likely to be a source of bugs.

### 2.5. Repository maintenance

A component repository once is created requires continuous maintenance. New components, as and when created have to be entered into the repository. The faulty components have to be tracked. Further, when new applications emerge, the order applications become obsolete. In this case, the obsolete components might have to be removed from the repository

## 3. Reuse at organization level

Reusability should be a standard part in all software development activities including specification, design, implementation, test, etc. ideally, there should be a steady flow of reusable components. In practice, however, things are not so simple. Extracting reusable components from projects that were completed in the past presents an important difficulty not encountered while extracting a reusable component from an ongoing project-typically; the original developers are no longer available for consultation. Development of new systems leads to an assortment of products, since reusability ranges from items whose reusability is immediate to those items whose reusability is highly improbable.

Achieving organization-level reuse requires adoption of the following steps:

- Assess of an item's potential for reuse
- Refine the item for greater reusability
- Enter the product in the reuse repository

We elaborate these three steps required to achieve organization-level reuse. Assessing a product's potential for reuse.

Reusability could act as the major tool in growth of small scale industry. Due to boost in globalization in technology, agile practices comes into act and many small scales companies are also adopting the Agile approach so better utilization of resources is major challenge as well as a opportunity for small scale industry [6].

## 4. Reuse advantages

#### 4.1 Increased dependability

Reused software that has been tried and tested in working systems should be more dependable than new software. The initial use of the software reveals any design and implementation faults. These are then fixed, thus reducing the number of failures when the software is reused.

#### 4.2 Reduced process risk

If software exists, there is less uncertainly in the costs of reusing that software than in the costs of development. This is an important factor for project management as it reduces the margin of error in project cost estimation. This is particularly true when relatively large software components such as sub-system are reused.

#### 4.3 Effective use of specialists

Instead of application specialists doing the same work on different projects, these specialists can develop reusable software that encapsulate their knowledge.

### 4.4 Standards compliance

Some standards such as user interface standards can be implemented as a set of standard reusable components. For example, if menus in a user interfaces are implemented using reusable components, all applications present the same menu formats to users. The use of standard user interfaces improves dependability as users. The use of standard user interfaces improves dependability as users are less likely to make mistakes when presented with a familiar interface.

## 5. Reuse problems

### 5.1 Increased maintenance costs

If the source code of a reused software system or component is not available then maintenance costs may be increased as the reused elements of the system may become increasingly incompatible with system changes.

### 5.2 Lack of tool support

CASE tool sets may not supports development with reuse. It may be difficult or impossible to integrate these tools with a component library system. The software process assumed by these tools may not take reuse into account.

### 5.3 Not-invented-here syndrome

Some software engineers sometimes prefer to re-write components as they believe that they can improve on the reusable component. This is partly to do with trust and partly to do with the fact that writing original software is seen as more challenging than reusing other people's software.

### 5.4 Creating and maintaining a component library

Populating a reusable component library and ensuring the software developers can use this library can be expensive. Our current techniques for classifying, cataloguing and retrieving software components are immature.

### 5.5 Finding, understanding and adapting reusable components

Software components have to be discovered in a library, understood and, sometimes, adapted to work in a new environment. Engineers must be reasonably confident of finding a component search as part of their normal development process.

## 6. Conclusion

There is an ample scope of research in stated area. Present study will reflect the importance of user requirements in any software project and how we can utilize resources with better approach with reusability. It will explore how way of requirement gathering is modified when reuse of recourses are taken into part at very initial step. It provides guidance for small scale companies to achieve growth during resource reusability.

## 7. References

[1] B.Jalender, Dr. A Govardhan, Dr.P Premchand," A Pragmatic approach to software reuse"; journal of Theoretical and Applied Information Technology.

[2] [Morisio02] Morision, M., Ezran, M., Tully, C.: Success and Failures in Software Reuse.IEEE Trans. Software Engineering, 28(4), pp. 340-357, April 2002.

[3] W.B. Frakes, C.J. Fox, "Sixteen Questions about Software Reuse", Comm. ACM, 38(6):75-87, 1995.

[4] Francisco J. Pino Æ Fe´lix Garcı´a Æ Mario Piattini, "Software process improvement in small and medium software enterprises: a systematic review": Published online: 21 November 2007_ Springer Science Business Media, LLC 2007.

[5] [Griss95] Griss, M.L., Wosser, M.: Making Reuse Work in Hewlett-Packard. IEEE Software, 12(1), pp. 105-107, January 1995.

[6] Version one, 3rd Annual Survey: "The State of Agile Development", www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf, 2008.

[7] Tim Menzies, Justin S. Di Stefano," More Success and Failure Factors in Software Reuse", IEEE trans. soft. eng. vol. xx, no. y, month 2002.

[8] Fayad, M. E., Laitinen, M., & Ward, R. P. (2000). Software engineering in the small. Communications of the ACM, 43(3), 115–118.